

# On Storing Data Objects of Business Processes on Blockchain Channels

Julius Köpke<sup>1</sup>, Adnan Brdanin<sup>1</sup>

<sup>1</sup>Alpen-Adria-Universität Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria

## Abstract

Blockchain systems provide strong support for enforceability if all data relevant for transaction verification is stored on-chain. An example is the effective prevention of double spending in the bitcoin network. However, this approach has substantial drawbacks if privacy is a concern. One alternative in the enterprise context are architectures where access to the blockchain can be restricted to specific participants, and separate blockchains (channels) between subsets of participants can be established. While such architectures are promising for supporting privacy, there can still be trade-offs between the supported levels of enforcement and the degree of privacy that can be natively supported by those systems. In this paper, we follow a model-based approach, where privacy and enforceability requirements are explicitly modeled in business processes. We provide a detailed analysis of how different levels of data privacy affect the possible placement of data in channels on Hyperledger Fabric (HLF) and to what degree decisions over data with privacy requirements can be enforced by the built-in transaction verification mechanism. Finally, we present a prototype based on distributed oracles that allows us to enforce the correctness of decisions over data located on different channels.

## Keywords

Smart Contracts, Permissioned Blockchains, Channels, Enforceability, Business Processes, Privacy, Privity

## 1. Introduction

Blockchain platforms have gained interest in the Enterprise context over the previous years. This is witnessed by a large body of research on the model-driven engineering of blockchain based applications and the execution of business processes on blockchains [1, 2]. Blockchain platforms provide highly desirable properties for inter-organizational collaborations such as observability, immutability, availability, persistency, and they allow to execute transactions in low-trust environments without a trusted third party. Peers in the blockchain network will only accept a new block if all its transactions are valid. On public blockchains, this is typically achieved by replaying the transactions on each peer. This allows blockchains to guarantee the correctness of transactions in zero-trust environments (e.g., cryptocurrencies).

Second-generation blockchain platforms such as Ethereum [3] support Turing-Complete languages for defining custom transactions referred to as smart contracts. The term smart contract was originally coined by N. Szabo in [4] as the counterpart of traditional contracts

---

*PoEM'2022 Workshops and Models at Work Papers, November 23-25, 2022, London, UK*


✉ [julius.koepke@aau.at](mailto:julius.koepke@aau.at) (J. Köpke); [adnanbr@edu.aau.at](mailto:adnanbr@edu.aau.at) (A. Brdanin)

🌐 <https://www.aau.at/isys/> (J. Köpke)

🆔 0000-0002-6678-5731 (J. Köpke)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

enforced by hardware and software. Szabo proposed the design goals of observability, online enforceability, and privacy for smart contracts. *Observability* describes the possibility of each participant to observe each other's performance. *Online enforceability* aims at making a breach of the contract infeasible. *Privacy* in the context of smart contracts aims at limiting the spread of knowledge and control to the participants with a contractual need-to-know. In this sense, privacy represents privacy requirements of smart contracts. We discriminate between the original smart contracts and code on blockchains by using the term smart contract code for the latter.

To guarantee the correctness of a transaction on a blockchain, all data for verification must be available on-chain. This can conflict with privacy requirements (e.g., when data must not be publicly available). Permissioned blockchains such as Hyperledger Fabric (HLF) [5] allow developers to restrict access to the shared ledger to known participants. HLF does not require a replay of transactions on all nodes. Instead, the execute order validate approach is used where the participants who have to endorse (execute and validate) transactions can be defined via so-called endorsement policies. Furthermore, such systems may allow the usage of sub-blockchains (channels in HLF). A channel is a logically separate blockchain that is only shared between a subset of participants of the main chain. Since transactions are bound to a specific chain, cross-channel transactions are not supported. In addition, HLF supports so-called private data collections. Private data collections are a type of off-chain storage that is governed by the blockchain network. They have the additional benefit that data can be used within blockchain transactions. However, off-chain storage has a major implication: The data itself is not stored on the blockchain. This can negatively impact blockchain properties such as immutability, observability, and persistency. It should be noted that depending on given requirements, data privacy can be achieved in various ways. This includes the usage of off-chain data, data encryption, or storing only the digest of data on-chain. However, we argue that for many applications on-chain data storage is beneficial. An example is the need for a full, immutable trace of the evolution of data items for fulfilling data provenance requirements.

This paper is framed by the model-driven engineering of inter-organizational processes on blockchains. In particular, we assume that the required levels of privacy and enforceability are explicitly represented in process models. We then analyze, how different degrees of privacy can be implemented via channels and we will discuss the consequences for the blockchain based verification of data-based decisions. In particular, we aim in addressing the following research questions: **RQ1:** How do different levels of privacy requirements of data objects influence the possible placement of data in channels? **RQ2:** Which kinds of blockchain transactions are required to enforce the correctness of data-based decisions over data with privacy requirements? **RQ3:** How can the resulting transactions be implemented on HLF?

The remainder of the paper is organized as follows. In Sect. 2 we discuss related work. Sect.3 provides the preliminaries for the paper by defining the process meta-model. In Sect. 4 we analyze how channels can be used to support privacy requirements (RQ1). Sect. 5 shows what kinds of transactions are required for enforcing the correctness of decisions over data and introduces a classification of cross-channel transactions (RQ2). Sect. 6 presents an implementation on HLF, supporting privacy requirements via channels and enforceability requirements via distributed oracles (RQ3). Finally, Sect. 7 concludes the paper.

## 2. Related Work

Recently, numerous approaches explored the model-driven development of blockchain-based applications and the execution business processes on blockchains (see [2, 1] for recent surveys). The works on BPM on blockchains range from choreography monitoring [6] and enforcement [7, 8] over on-chain business process engines such as [9] to the collaborative management of models on blockchains [10]. While the execution of business processes on blockchain peers very well with the objectives of enforceability and observability of smart contracts, privacy/privacy is not natively solved. The work in [11] gives an overview of the aspects of confidentiality in business processes on blockchain, and discuss existing techniques to (partially) address this aspect. Notably, most of the existing approaches focus on public blockchain systems and ignore constraints on privacy or generally assume that all non-control-data is off-chain. Some approaches such as [12, 8] apply encryption on public blockchains. A typical pattern for the execution of business processes on blockchains is the translation of process models to smart contract code. Consequently, a process is executed by calling blockchain transactions. This approach can enforce the correctness of the control-flow. However, enforcing the correctness of decisions within a process is challenging, if input data is stored off-chain or is encrypted since blockchain transactions have no access to the data. In order to access off-chain data or encrypted on-chain data oracles [13] are required. Oracles come with the risk of introducing central entities. This problem can be reduced by using distributed oracles [14] instead. Additionally, non-interactive Zero Knowledge Proofs [15] allow to proof the correctness of a transaction without revealing private input data on-chain. Notably, [8] also covers implementations on HLF, where different choreography instances are separated by channels and message data is transferred privately via private data collections. However, to the best of our knowledge, no existing approach analyzes the model-driven development of blockchains applications where privacy requirements of business processes data are realized via channels in detail.

## 3. Process Meta Model

For our analysis, we model blockchain based business processes in form of block-structured inter-organizational business process models [16, 17]. We repeat the essential definitions here. While such models do not provide the full expressiveness of BPMN, they have a clear semantics and allow us to focus on the core problems. In the remainder of the paper we use the usual object-style dot notation to access components. E.g. for a tuple  $t = (a, b)$ , we write  $t.a$  for accessing  $t_a$ .

**Definition 3.1** (Process Model). A business process model is a tuple  $P = (N, E, D, A)$  with a set of nodes  $N$ , a set of data objects  $D$  and a set of participants  $A$ . Nodes are connected by a set of directed edges  $E$  forming a directed acyclic graph. For each node  $n$  we define  $n.type \in \{activity, xor-split, xor-join, and-split, and-join, business-rule-task\}$  to declare the node type.  $n.name$  is the label of the node,  $n.d^r \subseteq P.D$ , the set of data objects read and  $n.d^w \subseteq P.D$ , the set of data objects written, and  $n.a \in P.A$ , the actor executing the node.

Each edge  $(m, n) \in P.E$  describes a precedence constraint between nodes  $m \in P.N$  and node  $n \in P.N$ . There is one node without predecessor, called start node and one node without

successor called stop node. Nodes of type *xor-split* and *and-split* have exactly 2 successors, nodes of type *xor-join* and *and-join* have exactly 2 predecessors.  $DV \subset P.D$  is a set of Boolean decision variables. Each *xor-split* node  $x$  is located immediately after a node of type *business-rule-task*. The business rule task  $br$  for  $x$  may read data objects and writes to a unique decision variable  $dv$  ( $br.d^w = \{dv\}$ ), which is also assigned to the *xor-split* node ( $x.d^r = \{dv\}$ ). One of the outgoing edges of  $x$  is adorned with  $dv$ , the other with  $\neg dv$ , indicating which path is chosen at runtime. This decision variable of a *xor-block* is not modified by any other node except the corresponding business rule task.

The process model is block structured. Therefore, each split node is associated with exactly one join node such that each path originating in the split node to the end node includes the associated join node.  $\square$

During the execution of a process instance, business rule tasks assign values to their decision variables. These values result in either following the *true* or *false* branch of the corresponding gateway. An instance type of a process  $P^I$  is determined by an instantiation of decision variables  $I = \{(d_1, v_1), \dots, (d_n, v_n)\}$ , where  $d \in D$  and  $v$  is a Boolean value.  $P^I$  is a sub-graph of  $P$  where each *xor-split* node has exactly one successor, the one that matches the value of the corresponding decision variable in  $I$ . We define  $PI(P)$  as the set of all possible instantiations of decision variables.

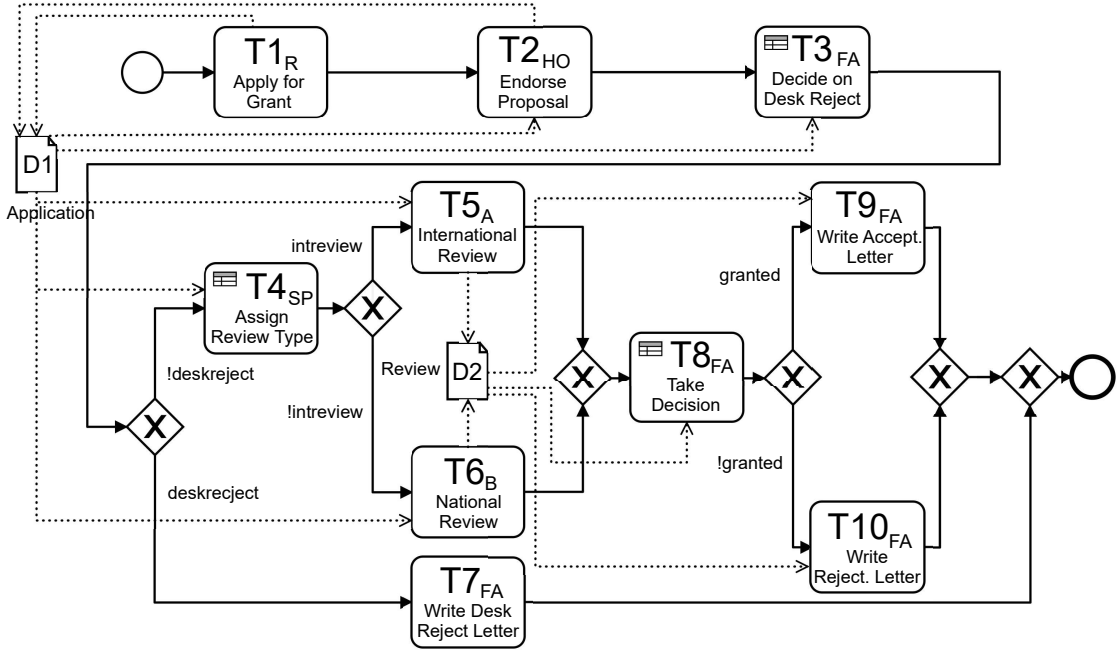
The *origin* for a data object  $d$  of a node  $n$  in an instance type  $P^I$ , denoted  $o(P^I, n, d)$ , is defined as a node  $m$  such that there exists a path  $p$  in  $P^I$  starting at  $m$  and ending at  $n$  and there is no step  $m'$  writing to  $d$  between  $m$  and  $n$  in  $P^I$ . A process model  $P = (N, E, D, A)$  is correct, iff for every instantiation  $I \in PI(P)$  of decision variables, for each input data object  $x \in D$  of each activity or business rule task  $n \in N$ :  $o(P^I, n, x)$  exists and is unique. The correctness criteria ensures that the process model is free of race-conditions of data objects.

### 3.1. Example Process

We present an example process first introduced in [17] in Fig. 1. It shows a collaboration between a researcher  $R$ , a host organization  $HO$ , a funding agency  $FA$ , a specialist  $SP$ , an international reviewing agency  $A$ , and a national reviewing agency  $B$ . First, in  $T1_R$  the researcher applies for a grant. This task stores the research proposal in data object  $D1$ . Then  $D1$  is processed and updated by the host organization in  $T2_{HO}$ . Next, the funding agency decides based on the provided data if the application is rejected due to formal reasons by setting the Boolean variable *deskreject*. In case of a rejection, a rejection letter is created in  $T7_{FA}$ . If the application is not rejected, an international  $A$  or national reviewing agency  $B$  is selected by setting *intreview* by a topic expert  $SP$  in  $T4_{SP}$ . During review, a review document  $D2$  is either created by  $T5_A$  or  $T6_B$ . The final decision is taken by the funding agency in  $T8_{FA}$  based on  $D1$  and  $D2$ . Finally, either an acceptance or rejection letter is created in  $T7_{FA}$ ,  $T9_{FA}$  or  $T10_{FA}$ .

### 3.2. Privity Spheres

Privity spheres were introduced in [12] and formalized in [17]. They allow to define which set of participants may gain access to which data object during process execution. We base our discussion on channels on privity spheres and therefore repeat the essential definitions of [17]



**Figure 1:** Example Collaboration between participants R, HO, FA, SP, A, B.

here. We annotate each data object  $d$  with its minimal sphere requirements  $d.sphere$ . The filler of the  $d.spheres$  can be a value in  $\{private, static, weak-dynamic, strong-dynamic\}$ .

**Definition 3.2** (Private Sphere). Let  $P$  be a process model. A participant is in the private sphere if she is an actor of the process:  $PrivateSphere(P) = P.A$   $\square$

**Definition 3.3** (Static Sphere). A participant  $a$  of a process is a member of the static sphere of some data object  $d$  if  $a$  is an actor of any task accessing  $d$  in  $P$ .  $StaticSphere(P, d) = \{a | a \in P.A : n \in P.N \wedge n.a = a \wedge (d \in n.d^w \vee d \in n.d^r)\}$

In the example process in Fig. 1, the static sphere for  $D1$  is  $\{R, HO, FA, SP, A, B\}$  since all participants execute at least one activity reading  $D1$ . The static sphere for  $D2$  is  $\{A, B, FA\}$ .

**Definition 3.4** (Weak-Dynamic Sphere). Let  $d \in P.D$  be a data object,  $w$  be a task writing to  $d$ . An actor  $a$  is in the weak-dynamic sphere of  $d$  for  $w$ , iff  $a$  is the actor of  $w$  or  $a$  is an actor of some task reading  $d$  where  $w$  is a possible origin for  $d$ :

$WeakDynamicSphere(P, d, w) = \{w.a\} \cup \{a | a \in P.A : n \in P.N \wedge n.a = a \wedge d \in n.d^r \wedge \exists I \in PI(P) : o(P^I, n, d) = w\}$   $\square$

In the example process in Fig. 1, the weak-dynamic sphere for  $D1$  for the writer  $T1_R$  is  $\{R, HO\}$  since only the  $HO$  can read the version written by  $R$ .

While the weak-dynamic sphere of a writer contains all participants, that can execute some task reading the written data value, the strong-dynamic sphere requires, that participants must certainly execute some tasks reading the data value.

**Definition 3.5** (Strong-Dynamic Sphere). Let  $d \in P.D$  be a data object,  $w$  be a task writing to  $d$ ,  $r$  be a node. An actor  $a$  is in the strong-dynamic sphere of  $w$  for  $d$  at node  $r$ , iff for every instance type where  $w$  is the origin of  $r$  for  $d$ ,  $a$  will execute some node reading the value of  $d$  from  $w$  or  $a$  is the actor of  $w$ .

$$\text{StrongDynamicSphere}(d, w, r) = \{a \mid a \in P.A : \exists I \in PI(P) : o(P^I, r, d) = w \wedge \forall I \in \{I \mid I \in PI(P) : o(P^I, r, d) = w\} \exists n \in P^I.N : (d \in n.d^r \wedge n.a = a \wedge o(P^I, n, d) = w) \vee a = w.a\} \quad \square$$

In the example process in Fig. 1, the strong-dynamic sphere for  $D1$  for the writer  $T1_R$  is  $\{R, HO\}$ . This equals the weak-dynamic sphere. However, the strong-dynamic sphere for the writer  $T2_{HO}$  is  $\{HO, FA\}$ . It does not contain  $A$  and  $B$  since it is not yet known if  $T5_A$  or  $T6_B$  will be executed. However, the strong-dynamic sphere for  $T2_{HO}$  for  $D1$  at position  $T5_A$  is  $\{HO, FA, SP, A\}$  since at this point it is known that  $A$  and  $SP$  will need the value of  $D1$ .

### 3.3. Modeling Enforceability Requirements of Decisions

Enforceability is an objective of smart contracts aiming to make a breach of the contract infeasible. However, the required degree of enforcement of the correctness of some decision may differ from decision to decision. While a decision whether an order is accepted may be taken solely by the buyer, a decision if the buyer has sufficient funds should be secured by the entire blockchain network. We assume that each business rule task  $dr$  of a process model has the additional property  $dr.verifiers$ . It holds a set of participants who have to verify the decision. This is well-aligned with endorsement policies of permissioned blockchains such as HLF. By definition, each verifier in  $dr.verifiers$  is a reader of all input data objects of the business rule task and therefore has access to them. Therefore, each verifier is in the strong-dynamic-sphere of all input data objects at the position of the business rule task.

## 4. Realizing Privacy Requirements via Channels

Since a channel is itself a sub blockchain, it is advisable to reuse channels for multiple instances of a collaboration between the same participants and not to create new channels for every new instance of a process. Otherwise, the channels are likely single block blockchains, questioning the basic principles of blockchains.

We now extend the process model from Def. 3.1 with channels: We define a channel as a tuple  $c = (id, t, M)$ , where  $id$  is a unique identifier and  $t$  can be *private* or *main* to indicate if  $c$  is a private channel or if it represents the main chain,  $M$  is a set of members representing the members of the channel ( $M \subset P.A$ ). We extend the definition of a process model from Definition 3.1 with an additional property  $C$  resulting in  $P = (N, E, D, A, C)$ .  $P.C$  is a set of channel definitions.

### 4.1. Example

We will now discuss how the different privacy requirements of data objects can be implemented using channels based on the example in Fig. 1. The resulting channels are shown in Figure 2.



If data objects  $D1$  and  $D2$  both require the *private* sphere, all data and the control flow state can be stored on the main chain. For the *static* sphere, every participant of the process owns at least one activity accessing  $D1$ . It can therefore be stored on the main chain.  $D2$  is only accessed by  $A, B, FA$ . Therefore,  $D2$  must be stored on a separate channel  $c2$  with members  $c2.M = \{A, B, FA\}$ . In the case of the *weak-dynamic* sphere, the value of  $D1$  written by  $T1_R$  must only be available to  $T2_{HO}$ . It is written to the private channel  $c1$  with  $c1.M = \{R, HO\}$ . The value of  $D1$  written by  $T2_{HO}$  must be available for all potential readers ( $FA, SP, A, B$ ). Therefore, it is written to a channel  $c2$  with  $c2.M = \{HO, FA, SP, A, B\}$ . For  $D2$  we have one channel  $c3$  shared between  $A$  and  $FA$  which is used by  $T5_A$ , if it is executed and a  $c4$  with  $c4.M = \{B, FA\}$  used by  $T6_B$  if it is executed. The *strong-dynamic* sphere is most challenging to realize via channels. As for the weak-dynamic case,  $R$  can store  $D1$  on a channel for  $R$ , and  $HO$  since  $T2_{HO}$  will always be executed. This differs for  $T2_{HO}$  since only  $T3_{FA}$  is certainly executed. Writing to a channel containing  $SP$  is only possible when the decision variable *dreject* evaluates to *false* in the continuation of the process. Consequently, we need 4 distinct channels to make the value of  $D1$  written by  $T2_{HO}$  available to the other participants:  $C1$  for  $HO, FA$ , unconditional.  $C2$  for  $HO, SP$  if *dreject* = *false*.  $C3$  for  $HO, A$  if *dreject* = *false* and *intreview* = *true*.  $C4$  for  $HO, A$  if *dreject* = *false* and *intreview* = *false*. None of the channels contains all participants. Therefore, no data object can be stored on the main chain.

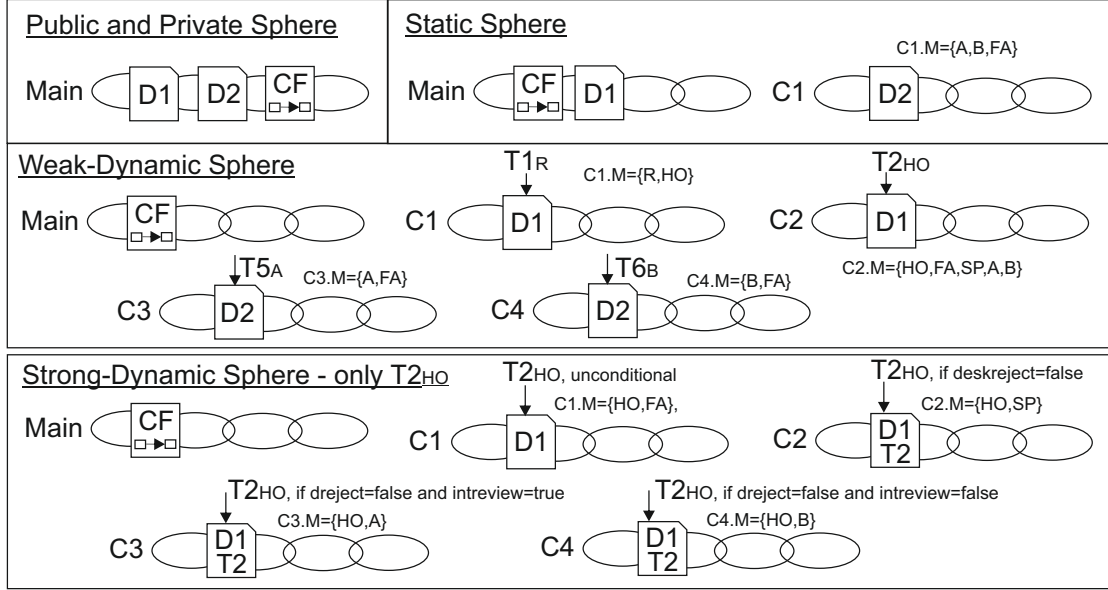
It should be noted that we cannot use one channel containing  $\{FO, FA, SP, A\}$  and another one for  $\{FO, FA, SP, B\}$  to store  $D1$  in  $T2_{HO}$  in the strong-dynamic case. At the time when  $T2_{HO}$  is executed it is not known if the application will be rejected by  $T3_{FA}$  and which reviewing agency will be selected in  $T4_{SP}$ . If we would only execute one process instance, and our target blockchain system supports the dynamic change of participants we could dynamically add participants to a single channel. In our scenario, where we aim in reusing channels for all process instances between the same participants, this is not possible.

## 4.2. Characterization of Implementations via Channels

We will now characterize the properties of possible implementations of privacy requirements via channels. For any implementation we can assume that it must be possible to know at any step in a process instantiation, from which channel a particular data object must be read by a particular actor. Based on our correctness criteria in Sect. 3, we know that the origin to read the data from is unique in every run of the process. Therefore, we can assume the existence of a function  $getChannel(r, d, a, I*_r)$ , that returns the channel, from which the participant  $a$  must read data object  $d$  at step  $r$  with the partial instantiation of decision variables  $I*_r$ .  $I*_r$  is defined as a subset of an instantiation  $I$  containing values for each decision variable of xor-split nodes on the path from start-node to  $r$ . We will now characterize the output of  $getChannel(r, d, a, I*_r)$  based on the privacy spheres of  $d$ .

**Proposition 4.1** (Placement Private). Let  $d \in P.D$  with  $d.sphere = private$ . Every participant  $a \in P.A$  can always read  $d$  from the main chain:

$$\{(id, 'main', P.A)\} = \{c | c \in P.C : c = getChannel(\_, d, \_, \_)\}.$$



**Figure 2:** Channel configuration for Example Process in Fig 1

**Proposition 4.2** (Placement Static). Every participant accessing  $d$  can always read  $d$  from the same channel:  $\{(id, type, member)\} = \{c | c \in P.C : c = getChannel(\_, d, \_, \_)\}$ , where  $member = \{a | a \in P.A : \exists t \in P.N \wedge t.a = a \wedge d \in t.d^r \cup t.d^w\}$  and if  $member = P.A$ , then  $type = 'public'$  otherwise  $type = 'private'$ .

**Proposition 4.3** (Placement Weak-Dynamic):. Let  $w_1$  and  $w_2$  be tasks writing to  $d$ . Tasks  $w_1$  and  $w_2$  must store  $d$  on different channels, if there are different sets of participants potentially reading  $d$  from  $w_1$  and from  $w_2$ . Due to conditional writes, also the channel, where  $d$  resides for a reading node  $r$  depends on the execution history of the process instance. Let  $a$  and  $b$  be actors in  $P.A$ . For the weak dynamic case,  $getChannel(r, d, a, P^{I^*r}) = getChannel(r, d, b, P^{I^*r})$  holds. E.g., the channel of  $d$  at step  $r$  only depends on  $P^{I^*r}$ .

**Proposition 4.4** (Placement Strong-Dynamic):. Due to conditional readers and only partially known decision outcomes during instance execution, a task  $w$  writing to some variable  $d$  may need to write it to multiple channels. Consequently, different participants may read the same value of  $d$  from different channels. Let  $a$  and  $b$  be actors. In contrast to the weak-dynamic case  $getChannel(r, d, a, P^{I^*r}) = getChannel(r, d, b, P^{I^*r})$  is only guaranteed if  $a = b$ .

**Prrof-Sketch 4.1** (of Prop. 4.1 - 4.4). The proofs directly follow from the definitions of the corresponding spheres in Def. 3.2 - 3.5.

## 5. Enforcing Correct Decisions via Transactions

After we have discussed how requirements on privacy expressed in form of privity spheres influence the location of data in channels, we now discuss the consequences for the enforcement



of data-based decisions. In order to take full advantage of the underlying blockchain system, a data-based decision should be realized by a single blockchain transaction. Therefore, it must be possible by the blockchain system to verify the correctness of the transaction. In the case of permissioned blockchains and in particular, on HLF, the transaction is replayed on the endorsing peers (e.g., for a business rule task  $d$  they are members of  $d.verifiers$ ). We base our discussion on the following assumptions: Like for many existing approaches for business processes on blockchains [9], we assume that process models are compiled into smart contract code. The smart contract code governing the control flow is deployed on the main chain. Business rule tasks of xor-split node are executed on-chain and publish their output on the main chain. This allows all participants to access all control-flow decisions to synchronize their processes.

## 5.1. Transaction Patterns

We now present patterns of the decision transactions for a business rule task  $c$  with a set of verifiers  $c.verifiers$  reading data objects  $D1$  and  $D2$  and executing some arbitrary decision expression  $exp$  and producing some Boolean output  $out$ .

**Private** In the private case we have one transaction where all input data and the output data are on the main chain (see Prop. 4.1).

```
read D1, D2 from main chain
out = exp(D1,D2)
publish out on main chain
```

**Static** In general, we cannot assume that  $D1$  or  $D2$  are located on the main chain. However, for every instantiation of the decision transaction, the data will be read from the same channels (see Prop.4.2). This leads to the following transaction with hard-coded channels:

```
read D1 from channel1
read D2 from channel2
out = exp(D1,D2)
publish out on main chain
```

**Weak-Dynamic** In contrast to the static case, the channels where  $D1$  and  $D2$  reside depend on the already taken decisions of the process  $I*_r$  (see Prop. 4.3) leading to the following transaction:

```
read D1 from getChannel(c,D1,_,I*c)
read D2 from getChannel(c,D2,_,I*c)
out = exp(D1,D2)
publish out on main chain
```

**Strong-Dynamic** In contrast to the weak-dynamic case, the location of data objects now additionally depends on the participant (see Prop. 4.4). Consequently different participants might read the data from different channels. This leads to the following transaction pattern for a participant  $p1$ :

```
read D1 from getChannel(c,D1,p1,I*c)
read D2 from getChannel(c,D2,p1,I*c)
out = exp(D1,D2)
publish out on main chain
```

It should be noted that each verifier and the actor of the business rule task may need to perform a different transaction since they potentially need to read the data objects from another channel.

## 5.2. Classification of Cross-Channel Transactions

Based on the previously described transactions for executing decisions, we introduce a classification of the required cross-channel transactions.

1. *None* All input and output data resides on the main chain, or the input is transaction input. Transaction output is also placed on the main chain.
2. *Simple* Input data is statically placed on one or more channels, and output data is written to the main chain.
3. *Complex* Input data is placed on one or more channels. In contrast to the static case, the channels where the data resides depend on the history of the process instance. Output data is statically written to the main chain.
4. *Dynamically Complex* Input data is placed on one or more channels; the channels where the data must be read from depends on the history of the process instance and additionally on the participant. Consequently, the participant executing the transaction and each verifier may need to read the input data from different channels.

To the best of our knowledge, no permissioned blockchain system currently supports cross-channel transactions. Therefore, only type *none* is directly supported by current blockchain platforms. Even *simple* cross-channel transactions are not supported as on-chain transactions. Therefore, we see the proposed classification as a benchmark for future blockchain platforms.

## 6. Implementation via Distributed Oracles

We have implemented a prototype<sup>1</sup> for the blockchain-based execution of business processes with privacy and enforceability requirements on HLF. For complying with privacy requirements, data is stored on separate channels. We assume that each participant operates its own blockchain node. The state of the control-flow of each process instance is stored on the main chain in form of a petri-net configuration. Therefore, control-flow transactions result in updates of the petri-net configuration. Process models are preprocessed in order to derive the *getChannel()* function from Sect. 4.2. Channels are reused between different process instances with the same assignments of participants to blockchain identities. In HLF, it is not possible to implement a single state-changing transaction reading data from different channels (cross channel transaction). However, since each decider and verifier has access to all required data, it is possible to validate

<sup>1</sup>Available online: <https://github.com/adnanb97/DistributedOracleHLF>. Implementation details are presented in [18]

decisions using oracles. To avoid the introduction of a central entity, we have implemented a distributed oracle. In our architecture, each peer executes a dedicated oracle service. Each such oracle service is triggered by specific changes of the ledger. Whenever some decision gets active, each oracle service of the verifying peers compute the decision locally and sends it digitally signed via HTTP to the oracle service of the deciding peer. The deciding peer waits until a configured quorum is reached and then issues one main chain transaction containing its own result and all signed votes as payload. The chain code behind this transaction checks if all signatures are correct and if the quorum was correctly reached. The correctness of this main chain transaction is enforced with the native endorsements of HLF. It should be noted that an (distributed) oracle is an off-chain component. As such the correctness of each member oracle is not guaranteed by the blockchain itself. However, due to the distribution of the oracle an attacker would need to attack multiple or even all participating oracle services depending on the required quorum to change the decision outcome.

## 7. Conclusion

Data privacy is a challenging problem on blockchains. Therefore, data is typically stored off-chain. However, this can have a negative impact on other requirements such as auditability, and immutability. For a large set of applications, permissioned blockchains can support privacy without threatening these properties for data storage. In this paper, we have analyzed how permissioned blockchain systems with channels can guarantee different levels of data privacy when data is stored on-chain (RQ1). Depending on the requirements, data must be stored statically or dynamically and potentially redundantly on different channels. We have then analyzed what kind of transactions are required for enforcing the correctness of data-based decisions when referenced data objects have privacy requirements (RQ2). Native implementations using the built-in verification mechanisms of current permissioned blockchain systems with channels are only possible with the lowest degree of privacy (private privacy sphere). All other levels lead to various kinds of cross-channel transactions. We have therefore introduced a classification of the resulting cross-channel transactions which is intended to be used as a reference for future permissioned blockchain platforms natively supporting cross-channel transactions. For the time being, we have implemented a prototype for enforcing the correctness of decisions over data on different channels via distributed oracles in HLF (RQ3). In this paper we have assumed that all verifying participants are allowed to read all input data. Interesting future work is to relax this assumption by using advanced cryptography such as Zero Knowledge Proofs[15] and to develop algorithms for the optimal placement of data objects in channels.

## References

- [1] S. Curty, F. Härer, H. Fill, Blockchain application development using model-driven engineering and low-code platforms: A survey, in: *Processings of EMMSAD and BPMDS 2022*, volume 450 of *LNBIP*, Springer, 2022, pp. 205–220.
- [2] F. Stiehle, I. Weber, Blockchain for business process enactment: A taxonomy and systematic literature review, in: *Business Process Management: Blockchain, Robotic Process*

Automation, and Central and Eastern Europe Forum, Springer International Publishing, Cham, 2022, pp. 5–20.

- [3] V. Buterin, Ethereum: A next-generation smart contract and decentralized application platform, 2014. URL: <https://ethereum.org/en/whitepaper/>, accessed: 2021-10-28.
- [4] N. Szabo, Formalizing and securing relationships on public networks., First Monday 9 (1997).
- [5] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proc. of EuroSys 2018, 2018, pp. 1–15.
- [6] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, J. Mendling, Untrusted business process monitoring and execution using blockchain, in: Proc. of BPM 2016, 2016, pp. 329–347.
- [7] F. Corradini, A. Marcelletti, A. Morichetta, A. Polini, B. Re, F. Tiezzi, Engineering trustable and auditable choreography-based systems using blockchain, ACM Trans. Manage. Inf. Syst. 13 (2022).
- [8] F. Corradini, A. Marcelletti, A. Morichetta, A. Polini, B. Re, E. Scala, F. Tiezzi, Model-driven engineering for multi-party business processes on multiple blockchains, Blockchain: Research and Applications 2 (2021) 100018.
- [9] O. Pintado, L. García-Bañuelos, M. Dumas, I. Weber, A. Ponomarev, Caterpillar: A business process execution engine on the ethereum blockchain, Software: Practice and Experience (2019).
- [10] H. Fill, F. Härer, Storing and attesting conceptual models on blockchains (invited paper), in: Comp. Proc. of Modellierung 2020, volume 2542, 2020, pp. 51–52.
- [11] B. Carminati, E. Ferrari, C. Rondanini, Blockchain as a platform for secure inter-organizational business processes, in: Proc. of CIC 2018, 2018, pp. 122–129.
- [12] J. Köpke, M. Franceschetti, J. Eder, Balancing privacy and enforceability of BPM-based smart contracts on blockchains, in: Proc. of BPM Blockchain Forum 2019, volume 361 of LNCS, Springer, 2019, pp. 87–102.
- [13] R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. Di Ciccio, I. Weber, M. Wöhrer, U. Zdun, Foundational oracle patterns: Connecting blockchain to the off-chain world, in: Proc. of BPM Blockchain and RPA Forum 2020, Springer, 2020, pp. 35–51.
- [14] D. Basile, V. Goretti, C. D. Ciccio, S. Kirrane, Enhancing blockchain-based processes with decentralized oracles, in: Business Process Management: Blockchain and RPA Forum - BPM 2021, volume 428 of LNBI, Springer, 2021, pp. 102–118.
- [15] O. Goldreich, Y. Oren, Definitions and properties of zero-knowledge proof systems, Journal of Cryptology 7 (1994) 1–32.
- [16] J. Köpke, M. Franceschetti, J. Eder, Optimizing data-flow implementations for inter-organizational processes, DAPD (2018) 1–45.
- [17] J. Köpke, M. Nečemer, Measuring the effects of confidants on privacy in smart contracts, in: Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum, Springer International Publishing, Cham, 2022, pp. 84–99.
- [18] A. Brdanin, Implementing Enforceability Requirements of Business Processes Using Hyperledger Fabric, Master's thesis, Alpen-Adria-Universität Klagenfurt, 2022.